

Chapter 2

Database System

Concepts and

Architecture

Chapter 2 Outline

- Data Models, Schemas, and Instances
- Three-Schema Architecture and Data Independence

Data Models, Schemas, and Instances

- **Data abstraction**
 - Suppression of details of data organization and storage
 - Highlighting of the essential features for an improved understanding of data

Data Models, Schemas, and Instances (cont'd.)

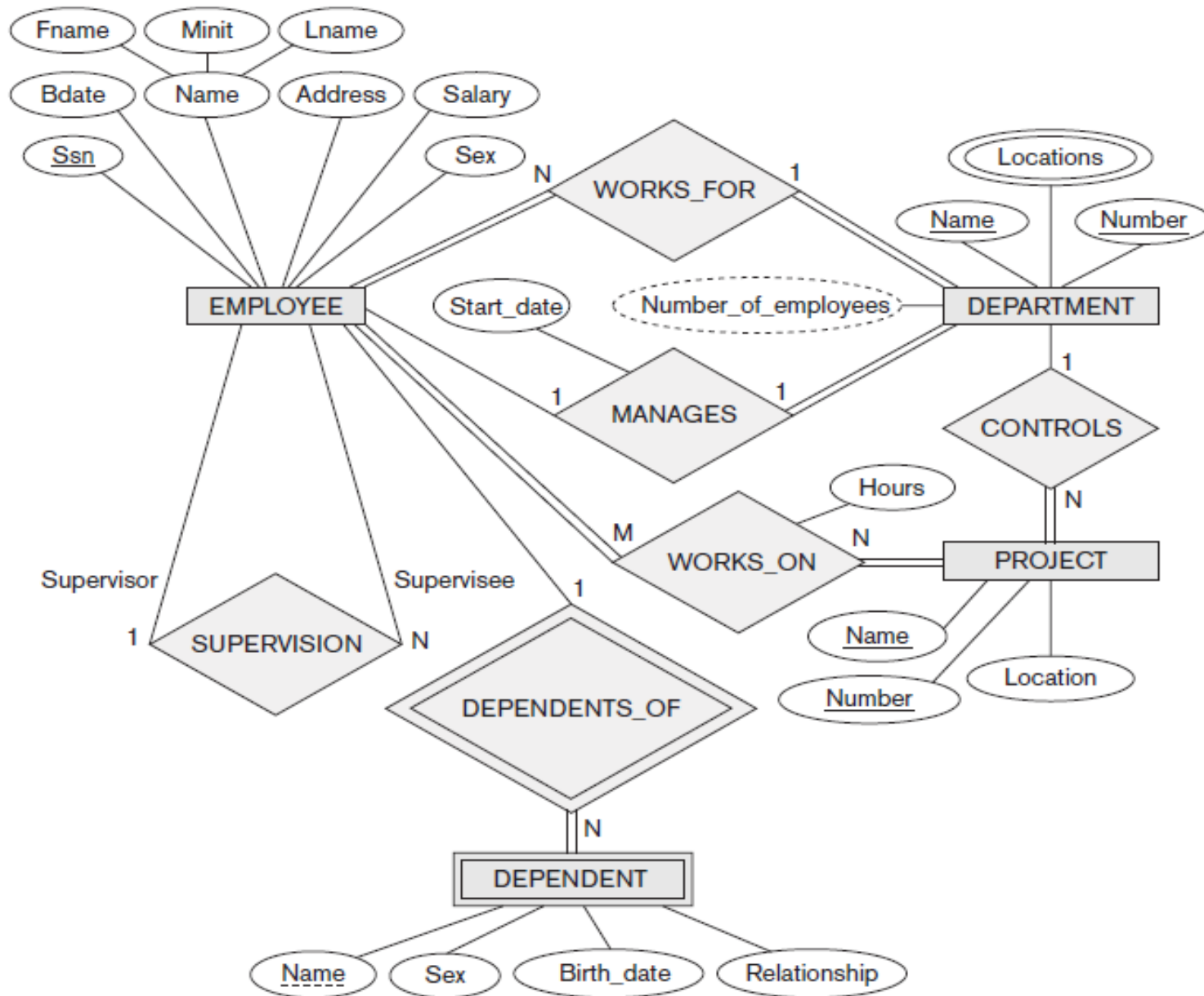
- **Data model**
 - Collection of concepts that describe the structure of a database
 - Provides means to achieve data abstraction
 - **Basic operations**
 - Specify retrievals and updates on the database
 - the **dynamic aspect** or **behavior** of a database application.
 - This allows the database designer to specify a set of valid user-defined operations that are allowed on the database objects. An example of a user-defined operation could be COMPUTE_GPA, which can be applied to a STUDENT object.

Categories of Data Models

- **High-level or conceptual data models**
 - Close to the way many users perceive data
- **Low-level or physical data models**
 - Describe the details of how data is stored on computer storage media
- **Representational data models**
 - Easily understood by end users
 - Also similar to how data organized in computer storage

High-level or conceptual data models

- **Entity**
 - Represents a real-world object or concept
- **Attribute**
 - Represents some property of interest
 - Further describes an entity
- **Relationship** among two or more entities
 - Represents an association among the entities
 - **Entity-Relationship model**



Categories of Data Models (cont'd.)

- **Relational data model**
 - Used most frequently in traditional commercial DBMSs
 - Represent data by using record structures and hence are sometimes called **record-based data models**.

EMPLOYEE

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	------------	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

PROJECT

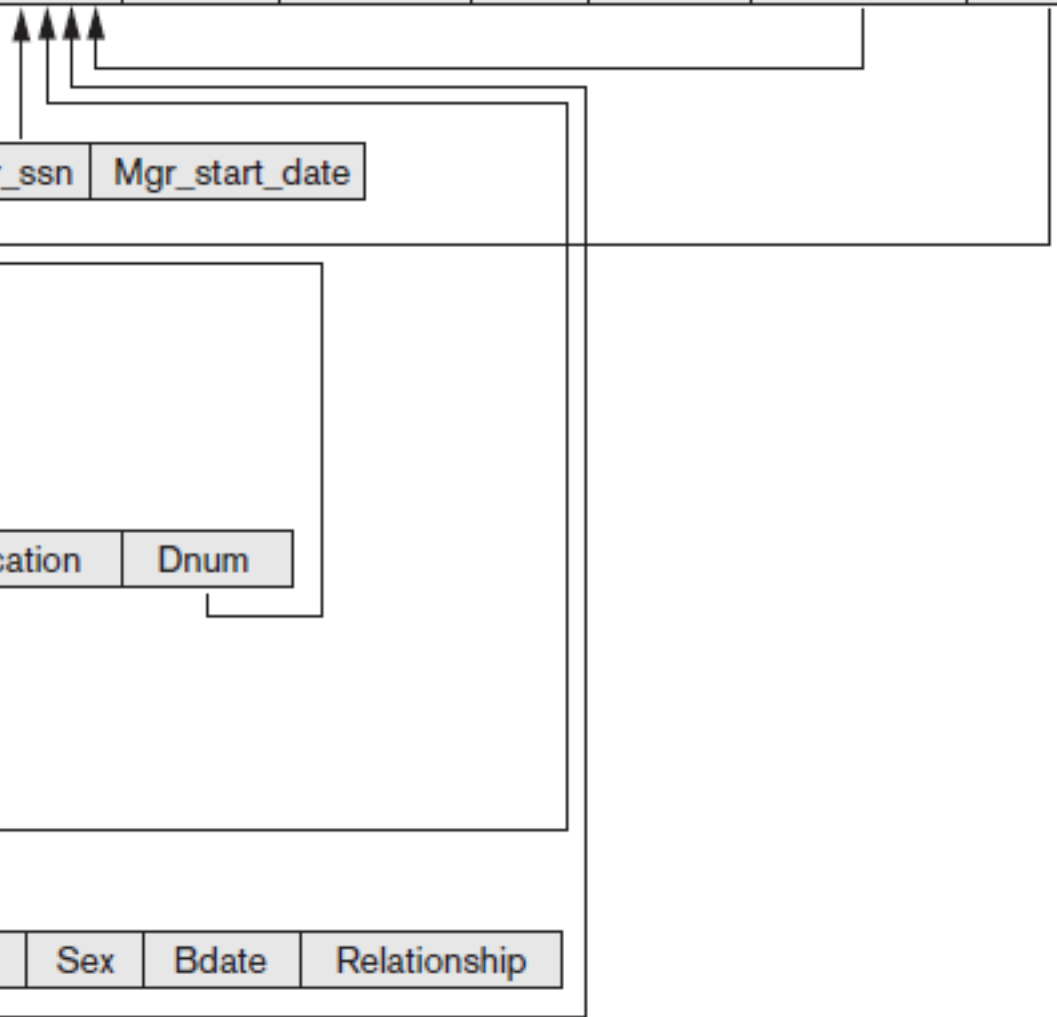
Pname	<u>Pnumber</u>	Plocation	Dnum
-------	----------------	-----------	------

WORKS_ON

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

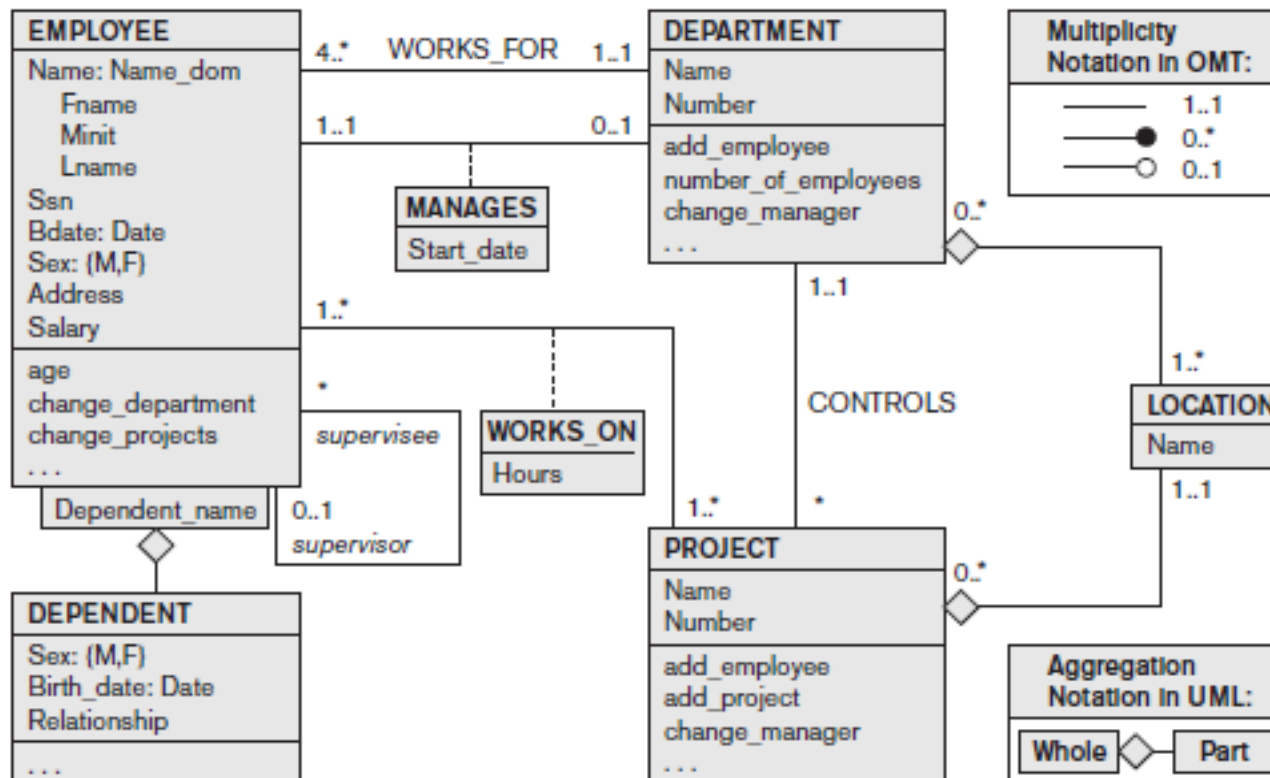
DEPENDENT

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------



Object data model

- New family of higher-level implementation data models Closer to conceptual data models (UML)



Categories of Data Models (cont'd.)

- **Physical data models**
 - Describe how data is stored as files in the computer by representing information such as record formats, record orderings, and access paths
 - **Access path**
 - Structure that makes the search for particular database records efficient
 - **Index**
 - Example of an access path
 - Allows direct access to data using an index term or a keyword

Schemas, Instances, and Database State

- **Database schema**
 - Description of a database which is specified during database design.
 - is not expected to change frequently
- **Schema diagram**
 - Displayed schema
 - The diagram displays the structure of each record type but not the actual instances of records.
- **Schema construct**
 - Each object in the schema such as STUDENT or COURSE
- **Database state or snapshot**
 - Data in database at a particular moment in time

Schemas, Instances, and Database State (cont'd.)

Figure 2.1

Schema diagram for the database in Figure 1.2.

STUDENT

Name	Student_number	Class	Major
------	----------------	-------	-------

COURSE

Course_name	Course_number	Credit_hours	Department
-------------	---------------	--------------	------------

PREREQUISITE

Course_number	Prerequisite_number
---------------	---------------------

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
--------------------	---------------	----------	------	------------

GRADE_REPORT

Student_number	Section_identifier	Grade
----------------	--------------------	-------

⁶Schema changes are usually needed as the requirements of the database applications change. Newer database systems include operations for allowing schema changes, although the schema change process is more involved than simple database updates.

⁷It is customary in database parlance to use *schemas* as the plural for *schema*, even though *schemata* is the proper plural form. The word *scheme* is also sometimes used to refer to a schema.

Database Schema vs. Database State

- Database State:
 - Refers to the **content** of a database at a moment in time.
 - The actual data stored in a database at a **particular moment in time**. This includes the collection of all the data in the database.
 - Also called database instance (or occurrence or snapshot).
 - The term *instance* is also applied to individual database components, e.g. *record instance*, *table instance*
- Initial Database State:
 - Refers to the database state when it is initially loaded into the system.
- Every time an update operation is applied to the database, we get another database state
- Valid State:
 - A state that satisfies the structure and constraints of the database.

Figure 5.6

One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Schemas, Instances, and Database State (cont'd.)

- **Schema evolution**
 - Changes applied to schema as application requirements change
- **Distinction**
 - The database schema changes very infrequently.
 - The database state changes every time the database is updated
- **Schema** is also called **intension**.
- **State** is also called **extension**

Example of a database state

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores
student and course
information.

Three-Schema Architecture and Data Independence

- The goal of the three-schema architecture is to separate the user applications from the physical database.
- **Internal level**
 - Describes physical storage structure of the database.
 - The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.
- **Conceptual level**
 - has a **conceptual schema**
 - Describes structure of the whole database for a community of users
 - The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.
- **External or view level**
 - Describes part of the database that a particular user group is interested in and hides the rest of the database from that user group.

Three-Schema Architecture and Data Independence (cont'd.)

External view 1

sNo	fName	lName	age	salary
-----	-------	-------	-----	--------

External view 2

staffNo	lName	branchNo
---------	-------	----------

Conceptual level

staffNo	fName	lName	DOB	salary	branchNo
---------	-------	-------	-----	--------	----------

Internal level

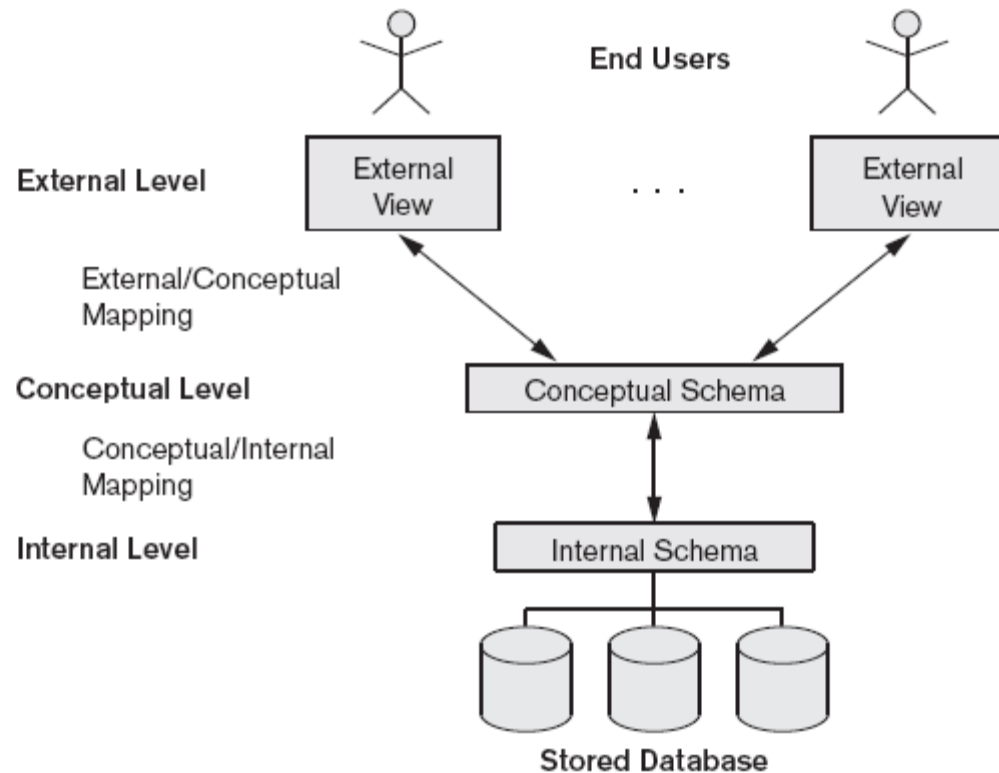
```
struct STAFF {  
    int staffNo;  
    int branchNo;  
    char fName [15];  
    char lName [15];  
    struct date dateOf Birth;  
    float salary;  
    struct STAFF *next;  
};  
index staffNo; index branchNo;
```

/* pointer to next Staff record */

/* define indexes for staff */

Three-Schema Architecture and Data Independence (cont'd.)

Figure 2.2
The three-schema architecture.

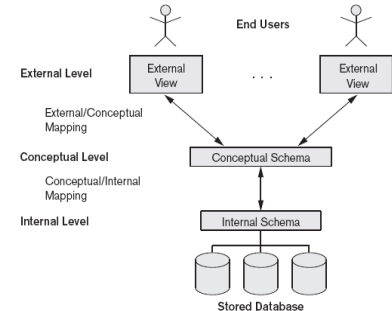


Three-Schema Architecture

- Mappings among schema levels are needed to transform requests and data.
 - Programs refer to an external schema, and are mapped by the DBMS to the internal schema for execution.
 - Data extracted from the internal DBMS level is reformatted to match the user's external view (e.g. formatting the results of an SQL query for display in a Web page)

Data Independence

Figure 2.2
The three-schema
architecture.



- Capacity to change the schema at one level of a database system
 - Without having to change the schema at the next higher level
- Types:
 - **Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs
 - **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well.

DBMS Languages

- **Data definition language (DDL)**
 - Defines both schemas
- **Storage definition language (SDL)**
 - Specifies the internal schema
- **View definition language (VDL)**
 - Specifies user views/mappings to conceptual schema
- **Data manipulation language (DML)**
 - Allows retrieval, insertion, deletion, modification

Centralized and Client/Server Architectures for DBMSs

- **Centralized DBMSs Architecture**
 - All DBMS functionality, application program execution, and user interface processing carried out on one machine

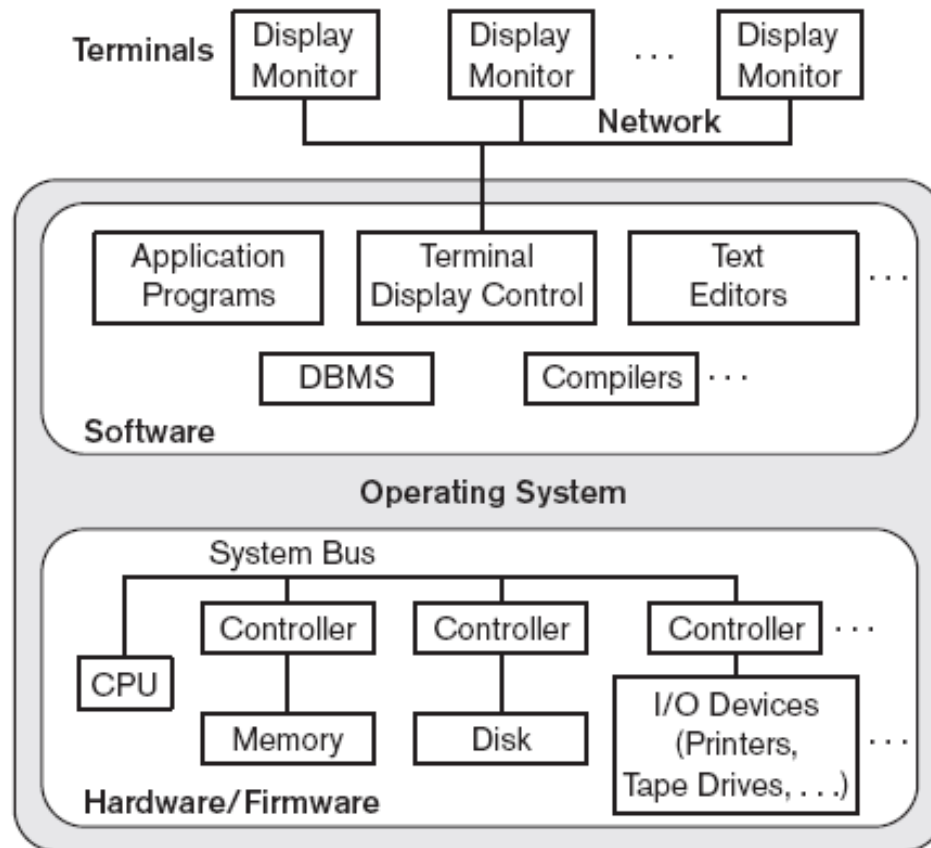


Figure 2.4
A physical centralized
architecture.

Basic Client/Server Architectures

- **Servers** with specific functionalities
 - **File server**
 - Maintains the files of the client machines.
 - **Printer server**
 - Connected to various printers; all print requests by the clients are forwarded to this machine
 - **Web servers** or **e-mail servers**

Basic Client/Server Architectures (cont'd.)

- **Client machines**
 - Provide user with:
 - Appropriate interfaces to utilize these servers
 - Local processing power to run local applications

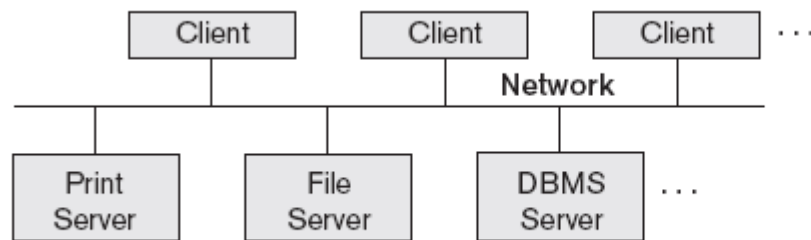


Figure 2.5
Logical two-tier
client/server
architecture.

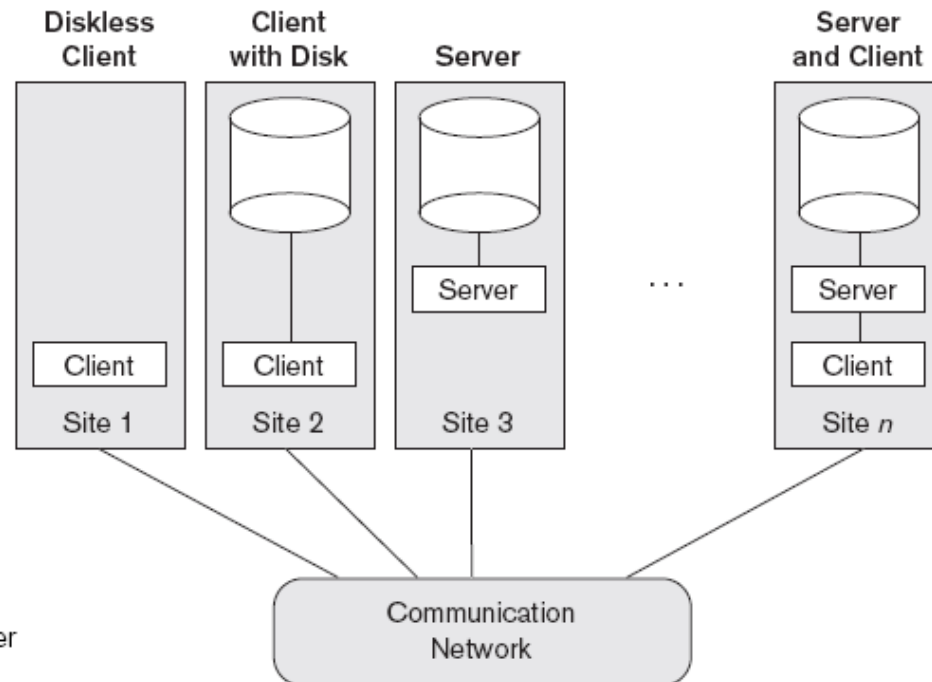


Figure 2.6
Physical two-tier client/server
architecture.

Basic Client/Server Architectures (cont'd.)

- **Client**

- User machine that provides user interface capabilities and local processing

- **Server**

- System containing both hardware and software
- Provides services to the client machines
 - Such as file access, printing, archiving, or database access

Two-Tier Client/Server Architectures for DBMSs

- Server handles
 - Query and transaction functionality related to SQL processing
- Client handles
 - User interface programs and application programs

Three-Tier and n-Tier Architectures for Web Applications

- **Application server or Web server**
 - Adds intermediate layer between client and the database server
 - Runs application programs and stores business rules
- **N-tier**
 - Divide the layers between the user and the stored data further into finer components

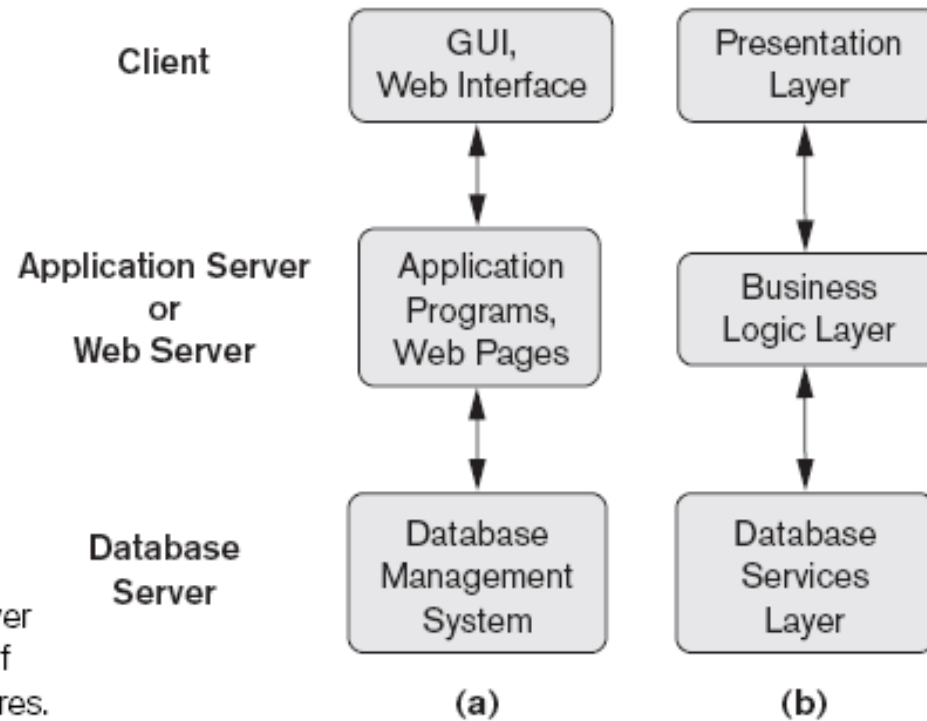


Figure 2.7

Logical three-tier client/server architecture, with a couple of commonly used nomenclatures.